RESONANCE

freename

Freename

Name Service Smart Contract Audit Report

Blockchain, Emerging Technology, and Web2
CYBERSECURITY PRODUCT & SERVICE ADVISORY

Document Control

PUBLIC **FINAL**(v2.0)

Audit_Report_FRNM-PRO_FINAL_20

Dec 4, 2023	v0.1	João Simões: Initial draft
Dec 5, 2023	v0.2	João Simões: Added findings
Dec 11, 2023	v0.3	João Simões: Added findings
Dec 12, 2023	v1.0	Charles Dray: Approved
Feb 6, 2023	v1.1	João Simões: Reviewed findings
Mar 18, 2023	v2.0	Charles Dray: Published

Points of Contact	Federico Costa	Freename	federico.costa@freename.io
	Charles Dray	Resonance	charles@resonance.security
Testing Team	Ilan Abitbol	Resonance	ilan.abitbol@resonance.security
	João Simões	Resonance	joao.simoes@resonance.security
	Michał Bazyli	Resonance	michal.bazyli@resonance.security

Copyright and Disclaimer

Contents

Executive Summary

Freename contracted the services of Resonance to conduct a comprehensive security audit of their smart contracts **between November 21, 2023** and **December 5, 2023**. The primary objective of the assessment was to identify any potential security vulnerabilities and ensure the correct functioning of smart contract operations.

During the engagement, Resonance allocated **3** engineers to perform the security review. The engineers, including an accomplished professional with extensive proficiency in blockchain and smart-contract security, encompassing specialized skills in advanced penetration testing, and in-depth knowledge of multiple blockchain protocols, devoted **11 days** to the project. The project's test targets, overview, and coverage details are available throughout the next sections of the report.

The ultimate goal of the audit was to provide **Freename** with a detailed summary of the findings, including any identified vulnerabilities, and recommendations to mitigate any discovered risks. The results of the audit are presented in detail further below.

System Overview

Freename is a domain name service that provides customers the ability to build their own Web3 domain ecosystem and become a registrar themselves without incurring additional fees. Customers may be able to buy and mint top-level and second-level domains and earn royalties for selling the rights to use their namespace. The protocol is deployed on Polygon, BNB SmartChain and Aurora.

The system is composed of three essential components: the registry where the top-level and second-level domains are minted and registered on the blockchain, the minting manager which encompasses the access control for the minting capabilities, and a proxy reader contract that is implemented to serve the purpose of being an explorer of the domain registry.

As a general workflow, an account with minting capabilities mints a top-level or second-level domain that is then registered on the registry for a specific owner. The ownership of a domain is guaranteed with the usage of ERC721 features - NFT tokens.

Repository Coverage and Quality

This section of the report has been redacted by the request of the customer.

Target

The objective of this project is to conduct a comprehensive review and security analysis of the smart contracts that are contained within the specified repository.

The following items are included as targets of the security assessment:

- Repository: `FreenameDomains/freename-solidity-ns/contracts`
- Hash: 4e5a833a8030a5bf5f88bfe7d0027259b4d2bb13

The following items are excluded:

- External and standard libraries
- Files pertaining to the deployment process
- Files inside `legacy/` directory

Methodology

In the context of security audits, Resonance's primary objective is to portray the workflow of a real-world cyber attack against an entity or organization, and document in a report the findings, vulnerabilities, and techniques used by malicious actors. While several approaches can be taken into consideration during the assessment, Resonance's core value comes from the ability to correlate automated and manual analysis of system components and reach a comprehensive understanding and awareness with the customer on security-related issues.

Resonance implements several and extensive verifications based off industry's standards, such as, identification and exploitation of security vulnerabilities both public and proprietary, static and dynamic testing of relevant workflows, adherence and knowledge of security best practices, assurance of system specifications and requirements, and more. Resonance's approach is therefore consistent, credible and essential, for customers to maintain a low degree of risk exposure.

Ultimately, product owners are able to analyze the audit from the perspective of a malicious actor and distinguish where, how, and why security gaps exist in their assets, and mitigate them in a timely fashion.

Source Code Review - Solidity EVM

During source code reviews for Web3 assets, Resonance includes a specific methodology that better attempts to effectively test the system in check:

1. Review specifications, documentation, and functionalities
2. Assert functionalities work as intended and specified
3. Deploy system in test environment and execute deployment processes and tests
4. Perform automated code review with public and proprietary tools
5. Perform manual code review with several experienced engineers
6. Attempt to discover and exploit security-related findings
7. Examine code quality and adherence to development and security best practices
8. Specify concise recommendations and action items
9. Revise mitigating efforts and validate the security of the system

Additionally and specifically for Solidity EVM audits, the following attack scenarios and tests are recreated by Resonance to guarantee the most thorough coverage of the codebase:

- Reentrancy attacks
- Frontrunning attacks
- Unsafe external calls
- Unsafe third party integrations
- Denial of service
- Access control issues

- Inaccurate business logic implementations
- Incorrect gas usage
- Arithmetic issues
- Unsafe callbacks
- Timestamp dependence
- Mishandled panics, errors and exceptions

Severity Rating

Security findings identified by Resonance are rated based on a Severity Rating which is, in turn, calculated off the **impact** and **likelihood** of a related security incident taking place. This rating provides a way to capture the principal characteristics of a finding in these two categories and produce a score reflecting its severity. The score can then be translated into a qualitative representation to help customers properly assess and prioritize their vulnerability management processes.

The **impact** of a finding can be categorized in the following levels:

1. Weak - Inconsequential or minimal damage or loss
2. Medium - Temporary or partial damage or loss
3. Strong - Significant or unrecoverable damage or loss

The **likelihood** of a finding can be categorized in the following levels:

1. Unlikely - Requires substantial knowledge or effort or uncontrollable conditions
2. Likely - Requires technical knowledge or no special conditions
3. Very Likely - Requires trivial knowledge or effort or no conditions

		Likelihood		
		Very Likely	Likely	Unlikely
Impact	Strong	Critical	High	Medium
	Medium	High	Medium	Low
	Weak	Medium	Low	Info

Repository Coverage and Quality Rating

The assessment of Code, Tests, and Documentation coverage and quality is one of many goals of Resonance to maintain a high-level of accountability and excellence in building the Web3 industry. In Resonance it is believed to be paramount that builders start off with a good supporting base, not only development-wise, but also with the different security aspects in mind. A product, well thought out and built right from the start, is inherently a more secure product, and has the potential to be a game-changer for Web3's new generation of blockchains, smart contracts, and dApps.

Accordingly, Resonance implements the evaluation of the code, the tests, and the documentation on a score **from 1 to 10** (1 being the lowest and 10 being the highest) to assess their quality and coverage. In more detail:

- Code should follow development best practices, including usage of known patterns, standard libraries, and language guides. It should be easily readable throughout its structure, completed with relevant comments, and make use of the latest stable version components, which most of the times are naturally more secure.
- Tests should always be included to assess both technical and functional requirements of the system. Unit testing alone does not provide sufficient knowledge about the correct functioning of the code. Integration tests are often where most security issues are found, and should always be included. Furthermore, the tests should cover the entirety of the codebase, making sure no line of code is left unchecked.
- Documentation should provide sufficient knowledge for the users of the system. It is useful for developers and power-users to understand the technical and specification details behind each section of the code, as well as, regular users who need to discern the different functional workflows to interact with the system.

Findings

During the security audit, several findings were identified to possess a certain degree of security-related weaknesses. These findings, represented by unique IDs, are detailed in this section with relevant information including Severity, Category, Status, Code Section, Description, and Recommendation. Further extensive information may be included in corresponding appendices should it be required.

An overview of all the identified findings is outlined in the table below, where they are sorted by Severity and include a **Remediation Priority** metric asserted by Resonance's Testing Team. This metric characterizes findings as follows:

- **"Quick Win"** Requires little work for a high impact on risk reduction.
- **"Standard Fix"** Requires an average amount of work to fully reduce the risk.
- **"Heavy Project"** Requires extensive work for a low impact on risk reduction.

RES-01	Bypass MintingManager To Mint On FNSRegistry		Resolved
RES-02	Previous Owner Of MintingManager May Takeover Contract		Resolved
RES-03	Signature Replay Attacks Possible On BaseForwarder		Resolved
RES-04	Impossibility Of Modifying _mintingManager And _propertyManager State Variables		Resolved
RES-05	Possible To Reset Any Token's Records		Resolved
RES-06	Possible Frontrunning Of initialize() Function		Resolved
RES-07	Missing _disableInitializers() On Upgradeable Contracts		Resolved
RES-08	Missing Validation Of Input Parameters On _setManyRecords() And _setManyRecordsByHash()		Resolved
RES-09	Missing Unblock Feature On Blocklist		Resolved
RES-10	Concurrent Usage Of Ownable And AccessControl		Acknowledged
RES-11	Possible To Mint tokenId 0 Leads To Undefined Behaviour		Resolved
RES-12	Minting SLD Does Not Require Existent TLD		Acknowledged
RES-13	Incorrect Implementation Of supportsInterface()		Resolved
RES-14	Overuse Of _presetOf() Function Calls		Resolved

RES-15	Missing Validation Of Input Parameters On _setRecord() Functions		Resolved
RES-16	Redundant Code On _beforeMinting()		Acknowledged
RES-17	Redundant Code On Storage Contracts		Acknowledged
RES-18	Unnecessary Usage of EnumerableMap		Acknowledged
RES-19	Incorrect Implementation Of ProxyReader Upgradeable Contract		Resolved
RES-20	Transfer Operations Always Result In Clearing Default Domain		Resolved
RES-21	No Usage of OpenZeppelin's PausableUpgradeable Contract		Acknowledged
RES-22	Incorrect Usage Of _init() And _init_unchained()		Resolved
RES-23	Usage Of transfer() Function Not Recommended		Resolved
RES-24	Unused Functions		Resolved

Bypass MintingManager To Mint On FNSRegistry

Critical | **RES-FRNM-PRO01** | Business Logic | **Resolved**

Code Section

REDACTED

Description

REDACTED

Recommendation

REDACTED

Status

The issue has been fixed in a71733eedeec63495376846f736d046308f13e9a.

Previous Owner Of MintingManager May Takeover Contract

Critical | **RES-FRNM-PRO02** | Data Validation | **Resolved**

Code Section

REDACTED

Description

REDACTED

Recommendation

REDACTED

Status

The issue has been fixed in a71733eedeec63495376846f736d046308f13e9a.

Signature Replay Attacks Possible On BaseForwarder

Critical | **RES-FRNM-PRO03** | Data Validation | **Resolved**

Code Section

REDACTED

Description

REDACTED

Recommendation

REDACTED

Status

The issue has been fixed in 8af4939c3521b1989c72b0b47df1d0ceaa700bf7.

Impossibility Of Modifying _mintingManager And _propertyManager State Variables

High | **RES-FRNM-PRO04** | Business Logic | **Resolved**

Code Section

REDACTED

Description

REDACTED

Recommendation

REDACTED

Status

The issue has been fixed in a71733eedeec63495376846f736d046308f13e9a.

Possible To Reset Any Token's Records

High | **RES-FRNM-PRO05** | Access Control | **Resolved**

Code Section

REDACTED

Description

REDACTED

Recommendation

REDACTED

Status

The issue has been fixed in a71733eedeec63495376846f736d046308f13e9a.

Possible Frontrunning Of initialize() Function

High | **RES-FRNM-PRO06** | Transaction Ordering | **Resolved**

Code Section

REDACTED

Description

REDACTED

Recommendation

REDACTED

Status

The issue has been fixed in a71733eedeec63495376846f736d046308f13e9a.

Missing _disableInitializers() On Upgradeable Contracts

Medium | **RES-FRNM-PRO07** | Business Logic | **Resolved**

Code Section

REDACTED

Description

REDACTED

Recommendation

REDACTED

Status

The issue has been fixed in 20f5963c95bbb6839b9c07574e6b712fce6d40d7.

Missing Validation Of Input Parameters On _setManyRecords() And _setManyRecordsByHash()

Medium **RES-FRNM-PRO08** Data Validation **Resolved**

Code Section

REDACTED

Description

REDACTED

Recommendation

REDACTED

Status

The issue has been fixed in e03c526ae57d82b4a38cd328b7c4707c3c7463f9.

Missing Unblock Feature On Blocklist

Medium | **RES-FRNM-PRO09** | Business Logic | **Resolved**

Code Section

REDACTED

Description

REDACTED

Recommendation

REDACTED

Status

The issue has been fixed in 1fe59173bd82519416e26c22c4d7d94ede6cbac6.

Concurrent Usage Of Ownable And AccessControl

Medium **RES-FRNM-PRO10** Code Quality **Acknowledged**

Code Section

REDACTED

Description

REDACTED

Recommendation

REDACTED

Status

The issue was acknowledged by Freename's team. The development team stated "Ownable And AccessControl serve for similar purposes, but in this case we think it's better to not to modify the code because even if similar, they are used for different purposes".

Possible To Mint tokenId 0 Leads To Undefined Behaviour

Medium **RES-FRNM-PRO11** Data Validation **Resolved**

Code Section

REDACTED

Description

REDACTED

Recommendation

REDACTED

Status

The issue has been fixed in 5905a1c5298cfd38f962d5af3b52995be5331e5e.

Minting SLD Does Not Require Existent TLD

Medium **RES-FRNM-PRO12** Data Validation **Acknowledged**

Code Section

REDACTED

Description

REDACTED

Recommendation

REDACTED

Status

The issue was acknowledged by Freename's team. The development team stated "This will be a 'won't fix', because we need to have the ability to let people mint domains on tlds that can be minted on other blockchains".

Incorrect Implementation Of supportsInterface()

Low | **RES-FRNM-PRO13** | Data Validation | **Resolved**

Code Section

REDACTED

Description

REDACTED

Recommendation

REDACTED

Status

The issue has been fixed in 5c93b46cac8b72def58421c12bd529675c8cb7ea.

Overuse Of _presetOf() Function Calls

Low | **RES-FRNM-PRO14** | Gas Optimization | **Resolved**

Code Section

REDACTED

Description

REDACTED

Recommendation

REDACTED

Status

The issue has been fixed in a237e4cf21cfd6cd29aae427c89df4463e6bf5a4.

Missing Validation Of Input Parameters On _setRecord() Functions

Low | **RES-FRNM-PRO15** | Data Validation | **Resolved**

Code Section

REDACTED

Description

REDACTED

Recommendation

REDACTED

Status

The issue has been fixed in 313cfe180873602f37c1f2b0789163fa8a710df6.

Redundant Code On _beforeMinting()

Low | **RES-FRNM-PRO16** | Gas Optimization | **Acknowledged**

Code Section

REDACTED

Description

REDACTED

Recommendation

REDACTED

Status

The issue was acknowledged by Freename's team. The development team stated "We have revised the logic of the blocklist enable and disable features and decided to future-proof the scalability of the protocol and the isBlocked() function, leaving it with the extra call to the isBlocklistEnabled().".

Redundant Code On Storage Contracts

Low | **RES-FRNM-PRO17** | Code Quality | **Acknowledged**

Code Section

REDACTED

Description

REDACTED

Recommendation

REDACTED

Status

The issue was acknowledged by Freename's team. The development team stated "The duplicated code has been put into the KeyStorageLibrary library and now KeyStorage.sol and PropertyKeyStorage.sol are lighter. Although, we decided not to make further improvements for the moment (remove PropertyKeyStorage.sol and KeyStorage.sol)".

Unnecessary Usage of EnumerableMap

Low | **RES-FRNM-PRO18** | Code Quality | **Acknowledged**

Code Section

REDACTED

Description

REDACTED

Recommendation

REDACTED

Status

The issue was acknowledged by Freename's team. The development team stated "EnumerablebleMap is indeed an overkill, but 'considering that Modifying the variable in place would most likely break the contract due to the type change', it will be done only if it's worth it and if we are able to properly test contracts upgradeability".

Incorrect Implementation Of ProxyReader Upgradeable Contract

Low | **RES-FRNM-PRO19** | Code Quality | **Resolved**

Code Section

REDACTED

Description

REDACTED

Recommendation

REDACTED

Status

The issue has been fixed in ff88cad5f4a8a7695c6caed988fde83a5d35d9fe.

Transfer Operations Always Result In Clearing Default Domain

Low | **RES-FRNM-PRO20** | Data Validation | **Resolved**

Code Section

REDACTED

Description

REDACTED

Recommendation

REDACTED

Status

The issue has been fixed in 1717fe461818aa512debff58b4e65e91030915db.

No Usage of OpenZeppelin's PausableUpgradeable Contract

Info | **RES-FRNM-PRO21** | Code Quality | **Acknowledged**

Code Section

REDACTED

Description

REDACTED

Recommendation

REDACTED

Status

The issue was acknowledged by Freename's team. The development team stated "Not fixing for the moment because it would break upgradability".

Incorrect Usage Of _init() And _init_unchained()

Info	RES-FRNM-PRO22	Code Quality	Resolved

Code Section

```
REDACTED
```

Description

```
REDACTED
```

Recommendation

```
REDACTED
```

Status

The issue has been fixed in 89d480e853072b2a0964fc99b1d1290f9732ab1f.

Usage Of transfer() Function Not Recommended

Info | **RES-FRNM-PRO23** | Code Quality | **Resolved**

Code Section

REDACTED

Description

REDACTED

Recommendation

REDACTED

Status

The issue has been fixed in 77fa4ec0031900574bc219c7cbee6d6f2c4bfb7c.

Unused Functions

Info | **RES-FRNM-PRO24** | Code Quality | **Resolved**

Code Section

REDACTED

Description

REDACTED

Recommendation

REDACTED

Status

The issue has been fixed in 5468fe7dd72e32ed7a666756e52dc195be04e3a3.

Proof of Concepts

This section of the report has been redacted by the request of the customer.